Lawrence Berkeley National Laboratory report LBNL-63375

An Explicit Runge-Kutta Iteration for Diffusion in the Low Mach Number Combustion Code

Joseph F. Grcar

August 2007

Abstract

This report describes the implementation of a Runge-Kutta iteration both for mixture-averaged and for multicomponent diffusion with Dufour and Soret effects in the low Mach number combustion code.

Contents

1 Introduction **3**

2 Equations **3**
2.1 Conservation Equations 3
2.2 Flux Formulas 3
2.3 Sources 4

3 Algorithmic Changes **4**
3.1 Overview 4
3.2 Runge-Kutta Method 5
3.3 Timestep Selection 6
3.4 Formula for $\nabla \cdot \mathbf{v}$ 6

4 Software Changes **7**
4.1 Activation Keywords 7
4.2 Operator Procedures 7
4.3 Boundary Conditions and the Operator Procedures 8
4.4 Complete Implementation 9

5 Tests **9**
5.1 1D Problem 9
5.2 2D Problem 10

6 Timing **12**
6.1 Data 12
6.2 Performance Improvements 13

A Appendix. Notation **13**

B Appendix. Convective Derivative **14**

C Appendix. Temperature Equation **15**

D Appendix. Alternate Enthalpy Equation **16**

List of Figures

1 Species in a 1D hydrogen flame 10
2 Atomic hydrogen in a 2D hydrogen flame 11

List of Tables

1 Edge values for different boundary conditions 8
2 Flame speed in the 1D flame 10
3 Peak $X(\mathrm{H})$ in the 2D flame 12
4 Time steps in the calculations 12

1 Introduction

This report describes the implementation of a Runge-Kutta iteration for both mixture-averaged and multicomponent diffusion in the low Mach number combustion code, LMC. The multicomponent model includes Dufour and Soret effects. Following a description of the relevant transport equations, the algorithm for performing the diffusive step is explained, the software modifications are outlined, and some tests are discussed.

This report assumes familiarity with the algorithm of Day and Bell [3] and with its implementation in the LMC code. This implementation of a new diffusion algorithm and new diffusion models is only for the 2D version and only for Cartesian coordinates. The modifications to the LMC software are in the `LMC/MultiCompDiff` subdirectory.

2 Equations

2.1 Conservation Equations

Mass diffusion enters into the conservation equation for each species [2, p. 1917, eqn. 1]

$$\rho \frac{DY_k}{Dt} = \frac{\partial \rho Y_k}{\partial t} + \nabla \cdot \rho Y_k \mathbf{v} = -\nabla \cdot \mathbf{j}_k + \dot{\rho}_k \tag{1}$$

and into the conservation equation for enthalpy [2, p. 1917, eqn. 2]

$$\rho \frac{Dh}{Dt} = \frac{\partial \rho h}{\partial t} + \nabla \cdot \rho h \mathbf{v} = -\nabla \cdot \mathbf{j}_q \; . \tag{2}$$

The right side of equation (2) omits pressure, radiative, and viscous terms. *The notation is defined in Appendix A.*

2.2 Flux Formulas

The fluxes in the conservation equations depend on whether diffusion is treated by a simplified mixture-averaged model or a full multicomponent model. The diffusive flux vectors for the two models are

$$\mathbf{j}_k^{(\mathrm{Mix})} = -\rho \mathcal{D}_{k,mix} \nabla Y_k \tag{3}$$

$$\mathbf{j}_k^{(\mathrm{Multi})} = \Big(\sum_\ell -\rho Y_k \mathcal{D}_{k,\ell} \nabla X_\ell \Big) - \frac{\rho Y_k \theta_k}{T} \nabla T \; . \tag{4}$$

The mixture averaged formula is alternatively stated in terms of ∇X_k, see the discussion by Bongers and de Goey in [2, p. 1921]. They also give the multicomponent formula [2, p. 1918, eqns. 4, 5 and intervening text]. The final term in equation (4) is the **Soret effect**. The heat flux vectors for the two models are

$$\mathbf{j}_q^{(\mathrm{Mix})} = \Big(\sum_k h_k \mathbf{j}_k \Big) - \lambda \nabla T \tag{5}$$

$$\mathbf{j}_q^{(\mathrm{Multi})} = \Big(\sum_k h_k \mathbf{j}_k \Big) - \lambda' \nabla T - \sum_k p \theta_k \nabla X_k \tag{6}$$

where $\mathbf{j}_k = \mathbf{j}_k^{(\mathrm{Mix})}$ or $\mathbf{j}_k^{(\mathrm{Multi})}$, respectively. The multicomponent formula is given in [2, p. 1918, eqn. 3]. The final term in equation (6) the **Dufour effect**. The ∇X_k term in equations (4) and (6) can be supplemented by a pressure term and body forces that will be neglected here. The subroutine EGSLTDR5 [4, p. 22] in EGLIB calculates $\rho Y_k \mathcal{D}_{k,\ell}$ and $\rho\theta_k$, so for purposes of evaluation, equation (6) is rewritten as

$$\mathbf{j}_k^{(\mathrm{Multi})} = \Big(\sum_k h_k \mathbf{j}_k \Big) - \lambda' \nabla T - \sum_k \frac{p}{\rho} (\rho\theta_k) \nabla X_k \, .$$

2.3 Sources

The low Mach number combustion code originally used the mixture model. The combination of equations (1) and (3) agrees with Day and Bell's [3, p. 537, eqn. 2]

$$\frac{\partial \rho Y_k}{\partial t} + \nabla \cdot \rho Y_k \mathbf{v} = \nabla \cdot \rho \mathcal{D}_{i,mix} \nabla Y_k + \dot{\rho}_k \, ,$$

except they write $\dot{\omega}_k$ for $\dot{\rho}_k$. The combination of equations (2, 5) agrees with their [3, p. 537, eqn. 3]

$$\frac{\partial \rho h}{\partial t} + \nabla \cdot \rho h \mathbf{v} = \Big(\sum_k \nabla \cdot h_k \rho \, \mathcal{D}_{k,mix} \nabla Y_k \Big) + \nabla \cdot \lambda \nabla T \, . \tag{7}$$

The multicomponent fluxes in equations (4) and (6) are given by Bongers and de Goey, as cited above. Giovangigli [5, p. 159, eqns. 7.1.6–7] uses different notation for the fluxes, $\mathcal{F}_k = \mathbf{j}_k^{(\mathrm{Multi})}$ and $\mathcal{Q} = \mathbf{j}_q^{(\mathrm{Multi})}$, and he writes them in terms of "flux diffusion coefficients" $C_{k,\ell} = \rho Y_k \mathcal{D}_{k,\ell}$ [5, p. 23, eqns. 2.5.14]. The combination of equations (2) and (6) agrees with Giovangigli [5, p. 12, eqn. 2.3.16; p. 21, eqns. 2.5.2, 2.5.3].

3 Algorithmic Changes

3.1 Overview

The algorithm uses the diffusive terms of equations (1) and (2) in four places.

1. A Godunov method that depends on diffusive terms is used to form cell-centered values for $(\nabla \cdot \rho Y_k \mathbf{v})^{(n+\frac{1}{2})}$ and $(\nabla \cdot \rho h \mathbf{v})^{(n+\frac{1}{2})}$ at half timesteps [3, pp. 541–542, eqns. 13–14] .

2. A Crank-Nicolson method is used to solve equations (1) and (2) for ρY_k and ρh at time $n+1$ with fixed advective and reactive terms [3, p. 542, eqns. 16–17]. The Crank-Nicolson method is part of a nonlinear Gauss-Seidel method [3, p. 541, top].

3. A projection method is used to solve the underdetermined equation [3, pp. 537–538, eqns. 7–8]

   $$\nabla \cdot \mathbf{v}^{(n+1)} = \mathbf{S} + \text{regularization term}$$

   where $\mathbf{S}$ is a complicated expression for $\nabla \cdot \mathbf{v}$ that is evaluated with the best available data at the end of the timestep.

4. The preceding calculations are with respect to the grid at a given level. The diffusive and other fluxes calculated on each level are used to adjust the solutions on the coarser levels.

An earlier effort was unsuccessful in implementing a multicomponent model with the Gauss-Seidel and Crank-Nicolson methods in item 2. The present work therefore replaces those methods by an explicit Runge-Kutta formula. The Runge-Kutta method is implemented for both the mixture-averaged and the multicomponent diffusion models.

3.2 Runge-Kutta Method

The explicit midpoint rule or Heun's method is a standard ODE integration formula. To integrate $v' = f(v)$ it performs the steps:

1. $v_n' = f(v_n)$
2. ${v_{n+1}}^* = v_n + \Delta t\, v_n'$
3. $({v_{n+1}}^*)' = f({v_{n+1}}^*)$
4. $v_{n+1} = v_n + \Delta t\, \dfrac{v_n' + ({v_{n+1}}^*)'}{2}$

Stars, *, indicate provisional values (in this case at time $n+1$). John Bell suggested implementing this in the following manner.

1. The starting values at time n are:

   (a) the old state values,

   $$state_Y^{(n)} = (\rho Y_k)^{(n)} \qquad state_h^{(n)} = (\rho h)^{(n)}$$

   (b) the divergences of the advective fluxes at the half timestep,

   $$adv_Y^{(n+1/2)} = \nabla \cdot (\rho Y_k \mathbf{v})^{(n)} \qquad adv_h^{(n+1/2)} = \nabla \cdot (\rho h \mathbf{v})^{(n)}$$

   (c) the contribution from the chemistry over the first half time step,

   $$chem_Y^{(n+1/2)} = \int_{t_n}^{t_{n+1/2}} \dot{\rho}_k dt$$

   In addition, there are functions, or operators, that evaluate the divergences of the diffusive fluxes.

   $$\begin{aligned} \mathcal{L}_Y(\text{state data}) &= \nabla \cdot \mathbf{j}_k \quad \text{for all species } k \\ \mathcal{L}_h(\text{state data}) &= \nabla \cdot \mathbf{j}_q \end{aligned}$$

   These two operators $\mathcal{L}_Y$ and $\mathcal{L}_h$ collectively constitute the function f in the equations of the Runge-Kutta method.

2. Evaluate operators at time n.

   (a) $diff_Y^{(n)} = \mathcal{L}_Y(state_Y^{(n)} + chem_Y^{(n+1/2)})$

   (b) $diff_h^{(n)} = \mathcal{L}_h(state_h^{(n)})$

3. Form provisional values for time $n+1$.

   (a) $state_Y^{(n+1)*} = state_Y^{(n)} + chem_Y^{(n+1/2)} + adv_Y^{(n+1/2)} + \Delta t\, diff_Y^{(n)}$

   (b) $state_h^{(n+1)*} = state_h^{(n)} + adv_h^{(n+1/2)} + \Delta t\, diff_h^{(n)}$

4. Evaluate operators at provisional values for time $n+1$.

   (a) $diff_Y^{(n+1)*} = \mathcal{L}_Y(state_Y^{(n+1)*})$

   (b) $diff_h^{(n+1)*} = \mathcal{L}_h(state_h^{(n+1)*})$

5. Form values for time $n+1$.

   (a) $state_Y^{(n+1)} = state_Y^{(n)} + chem_Y^{(n+1/2)} + adv_Y^{(n+1/2)} + (\Delta t/2)\,(diff_Y^{(n)} + diff_Y^{(n+1)*})$

   (b) $state_h^{(n+1)} = state_h^{(n)} + adv_h^{(n+1/2)} + (\Delta t/2)(diff_h^{(n)} + diff_h^{(n+1)*})$

3.3 Timestep Selection

John Bell suggested the following limit on the diffusive time step for the Runge-Kutta method,

$$\Delta t \le c\,\frac{\Delta x^2}{2\,N\,\mathcal{D}_{k,mix}}\,, \tag{8}$$

where the coefficient is chosen conservatively as $c = 0.5$. Note that this formula depends on the mixture-averaged diffusion coefficients whether or not the mixture-averaged or multicomponent model is used in the algorithm. This time step limit is in addition to the CFL limit of the LMC code. For laminar flows, this limit should be more stringent.

3.4 Formula for $\nabla \cdot \mathbf{v}$

As explained in Day and Bell [3, p. 537], the constraint is obtained by taking the convective derivative of the equation of state written in the first of the following forms (the second equality is just the definition of the mean molecular weight)

$$p_0 = \rho RT \sum_k \frac{Y_k}{m_k} = \frac{\rho RT}{\bar{m}}\,.$$

The derivative is

$$0 = RT \sum_k \frac{Y_k}{m_k}\frac{D\rho}{Dt} + \rho R \sum_k \frac{Y_k}{m_k}\frac{DT}{Dt} + \rho RT \sum_k \frac{1}{m_k}\frac{DY_k}{Dt}$$

The equation of state simplifies all three terms to

$$0 = p_0 \frac{1}{\rho}\frac{D\rho}{Dt} + p_0 \frac{1}{T}\frac{DT}{Dt} + p_0 \sum_k \frac{\bar{m}}{m_k}\frac{DY_k}{Dt}$$

so that

$$-\frac{1}{\rho}\frac{D\rho}{Dt} = \frac{1}{T}\frac{DT}{Dt} + \sum_k \frac{\bar{m}}{m_k}\frac{DY_k}{Dt} \ .$$

Equation (10) removes the convective derivative of ρ,

$$\begin{aligned}
\nabla \cdot \mathbf{v} &= \frac{1}{T}\frac{DT}{Dt} + \sum_k \frac{\bar{m}}{m_k}\frac{DY_k}{Dt} \\
&= \frac{1}{\rho c_{p,mix} T}\left(-\nabla \cdot \mathbf{j}_q - \sum_k h_k \left(\dot{\rho}_k - \nabla \cdot \mathbf{j}_k\right)\right) \quad \text{from equation (12)} \\
&+ \frac{1}{\rho}\sum_k \frac{\bar{m}}{m_k}\left(\dot{\rho}_k - \nabla \cdot \mathbf{j}_k\right) \quad \text{from equation (1).}
\end{aligned}$$

Therefore

$$\begin{aligned}
\mathbf{S} &= \nabla \cdot \mathbf{v} \\
&= \frac{-1}{\rho c_{p,mix} T}\left(\nabla \cdot \mathbf{j}_q\right) + \sum_k \frac{1}{\rho}\left(\frac{h_k}{c_{p,mix}T} - \frac{\bar{m}}{m_k}\right)\left(\nabla \cdot \mathbf{j}_k - \dot{\rho}_k\right) \ . \qquad (9)
\end{aligned}$$

The simplification has been checked using Mathematica. This formula can be applied to both the mixture-averaged and the multicomponent fluxes.

4 Software Changes

4.1 Activation Keywords

In the `inputs` file, the keyword `ht.do_rk_diffusion = 1` (default `0`) selects the Runge-Kutta algorithm. With this option it is then the default to use the multicomponent model for diffusion. The additional keyword `ht.rk_mixture_averaged = 0` (default `1`) can be used to choose the mixture-averaged model instead. The coefficient $c = 0.5$ in the timestep equation (8) can be changed by the keyword `ht.rk_time_step_multiplier`.

4.2 Operator Procedures

The main part of the implementation of the Runge-Kutta method consists of a single procedure `rk_diffusion_operator` to evaluate the operators of the Runge-Kutta method in section 3.2. Both the fluxes $\mathbf{j}_k$ and $\mathbf{j}_q$ and their divergences are evaluated. This procedure is added to `LMC/HeatTransfer.cpp`.

The actual calculations are done in fortran subroutines `FORT_RK_MIXTURE_AVERAGED` and `FORT_RK_MULTICOMPONENT` for the mixture-averaged and multicomponent models, re-

spectively. These routines are added to `LMC/HEATTRANSFER_2D.F`. Inputs are `Fab`s in the `MultiFab`s for a given level of the new or old state, with one layer of ghost cells filled by a `FillPatchIterator`. Unlike much of the LMC software, these routines are self-contained. Transport properties are formed by calls from the fortran routines to CHEMKIN [6] or EGLIB [4] subroutines. Derivatives and edge values are formed in-line as needed.

4.3 Boundary Conditions and the Operator Procedures

The `FORT_RK_MIXTURE_AVERAGED` and `FORT_RK_MULTICOMPONENT` subroutines require information about boundary conditions to be able to evaluate values and derivative on cell edges that coincide with boundaries. The boundary condition array for a `Fab` is determined by the index of the `Fab` in its `MultiFab` and is accessed by `AmrLevel::getBCArray`. The array entry `bc(SDIM,2,ncomps)` gives the type of boundary for: a given axis, for the low or high (1 or 2) side, and for a given component in the state. The meanings of the numeric codes for the boundary condition types are described in `amrlib/BC_TYPES.H`.

The boundary conditions determine how the values and the derivatives of state variables should be obtained on cell edges that coincide with the boundary. This information is summarized in Table 1.

1. For internal boundaries, the `FillPatchIterator` places the value from the adjacent grid into the ghost cell. Edge values are obtained by averaging. Derivatives are obtained by centered divided differences.

2. For external Dirichlet boundaries, the `FillPatchIterator` places the value desired on the edge into the ghost cell. No averaging is needed, but derivatives must be obtained using one-sided, three-point formulas for second order accuracy. The low-side or left-side formula is

$$\left.\frac{dy}{dx}\right|_{-1/2} \approx \left(\frac{-8}{3\,\Delta x}\right) y_{-1/2} + \left(\frac{3}{\Delta x}\right) y_0 + \left(\frac{-1}{3\,\Delta x}\right) y_1 \ .$$

   where y is the variable, Δx is the grid spacing, and the subscripts refer to cell centered (integer subscripts) or to edge values. The high-side or right-side formula is

$$\left.\frac{dy}{dx}\right|_{1/2} \approx \left(\frac{1}{3\,\Delta x}\right) y_{-1} + \left(\frac{-3}{\Delta x}\right) y_0 + \left(\frac{8}{3\,\Delta x}\right) y_{1/2} \ .$$

3. For other boundaries, edge values should be obtained by averaging. Derivatives should be set to zero.

Table 1: *Evaluation of edge values and derivatives in the* `FORT_RK_MIXTURE_AVERAGED` *and* `FORT_RK_MULTICOMPONENT` *routines as determined by the type of boundary condition.*

case	boundary flag	edge value	edge derivative
1	internal Dirichlet	mean	centered, 2-point
2	external Dirichlet	ghost cell	1-sided, 3-point
3	other	mean	0

4.4 Complete Implementation

The `rk_diffusion_operator` procedure is accessed wherever diffusive fluxes or divergences are needed. To that end, the following changes are made to `LMC/HeatTransfer.cpp`, which respectively address the four items in section 3.1. These changes can be identified by searching for logical tests of the Boolean variable `do_rk_diffusion`.

1. The `getViscTerms` procedure is modified to call `rk_diffusion_operator` to obtain flux divergences of ρY_k and ρh.

2. The `advance` procedure is modified to perform the explicit Runge-Kutta method described in section 3.2.

3. The `calc_divu` procedure is modified to evaluate $\mathbf{S}$ by equation (9).

4. The fluxes underlying the calculation in item 2 are tabulated and stored in the data structure supplied by the existing multilevel synchronization algorithm. Data are entered using the existing procedures `CrseInit` and `FineAdd`. The calls to these routines are contiguous with the changes to the `advance` procedure for item 2.

   The flux corrections are applied in the `mac_sync` procedure. The more accurate fluxes on a fine grid are applied as corrections to the solution on the underlying coarse grid. In the original algorithm, since the diffusive contributions to the cell centered values are obtained implicitly, by the Crank-Nicolson method, the corrections are also obtained implicitly [3, p. 545, eqns. 18 and 19]. However, in the Runge-Kutta algorithm, since the diffusive contributions are obtained explicitly, the corrections are also explicit and affect only the coarse cells immediately surrounding the fine grid boundary. The original algorithm is structured in such a way that the explicit corrections can be applied by omitting the relevant Crank-Nicolson update to the coarse grid in the `mac_sync` procedure.

An additional change is needed to choose the Runge-Kutta timestep:

5. The `estTimeStep` procedure is expanded to select a timestep for the Runge-Kutta method by equation (8). That is done by the new procedure `rk_step_selection` in `LMC/HeatTransfer.cpp`, and by the new subroutine `FORT_RK_STEP_SELECTION` in `LMC/HEATTRANSFER_2D.F`.

5 Tests

5.1 1D Problem

The first test problem consists of a freely-propagating, planar, hydrogen-air flame with fuel equivalence ratio $\phi = 0.37$. The computational domain is $x \times y = 0.25 \times 4.0$ (cm). The grid has 8×128 cells at the coarsest level with two levels of $\times 2$ refinement. The fine grid resolution is therefore 78.125 (micron). The flame evolves over 0.05 (sec) during which it is steadied at $y = 2.0$ (cm) using the control algorithm discussed in [1].

Figure 1 displays the mole fractions of the species with respect to distance along the y axis. The CFL limit of 0.9 allowed rather large timesteps in the original LMC algorithm.

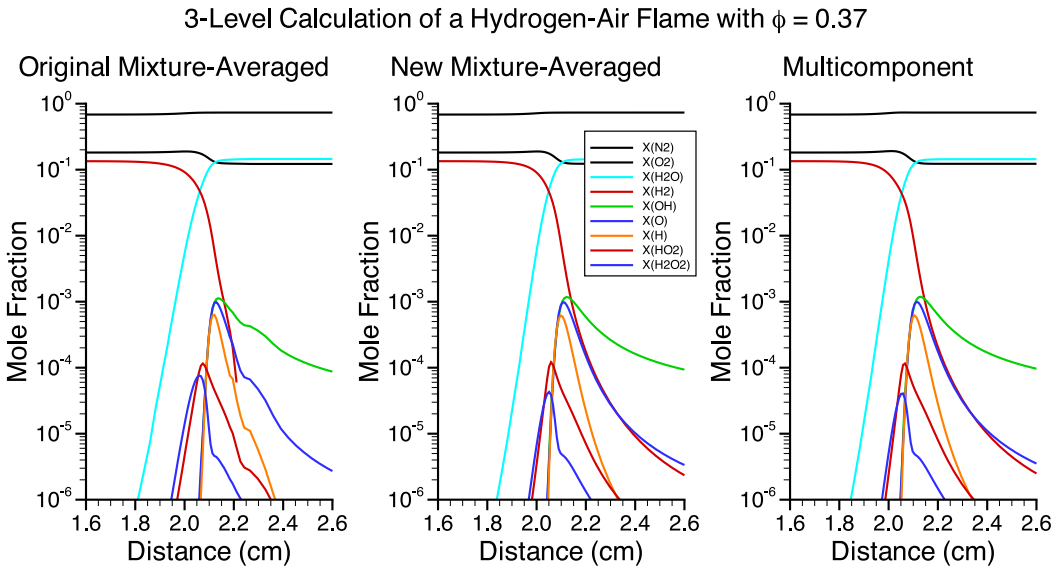

Figure 1: *Three calculations of mole fractions in the same freely-propagating, planar, hydrogen-air flame.*

As a result, the simulation performed by the original LMC algorithm is under-resolved in time. Table 2 lists the computed flame speeds. The speed predicted by the original LMC algorithm can be neglected because of the under-resolution.

Both Figure 1 and Table 2 show very little difference between the mixture-averaged and the multicomponent diffusion models. This agreement was also found by Bongers and de Goey [2, p. 1922, fig. 1, lower left panel]. They saw little or no difference between the two models except for rich flames or for flames burning in pure oxygen.

Table 2: *Three calculations of flame speed for the same freely-propagating, planar, hydrogen-air flame.*

diffusion model	solution algorithm	flame speed (cm / sec)
mixture-averaged	original LMC	12.133
mixture-averaged	Runge-Kutta	15.224
multicomponent	Runge-Kutta	15.153

5.2 2D Problem

The second test problem consists of a vortex encountering a freely-propagating, planar, hydrogen-air flame with fuel equivalence ratio $\phi = 0.37$. This problem resembles but is not identical to a test case of Day and Bell [3, pp. 546–549, figs. 1–3]. The computational domain is $x \times y = 1.2 \times 4.8$ (cm). The flame is again located at $y = 2.0$ (cm) and is steadied there this time by fixing an inflow velocity of 15 (cm/s). The grid has 48×192 cells at the

coarsest level with two levels of $\times 2$ refinement. The fine grid resolution is therefore 62.5 (micron).

Figure 2 displays the mole fraction of the hydrogen radical, $X(\mathrm{H})$, after 0.01 (s) of simulated time. The CFL limit in this calculation is 0.80 so the flame again may not be

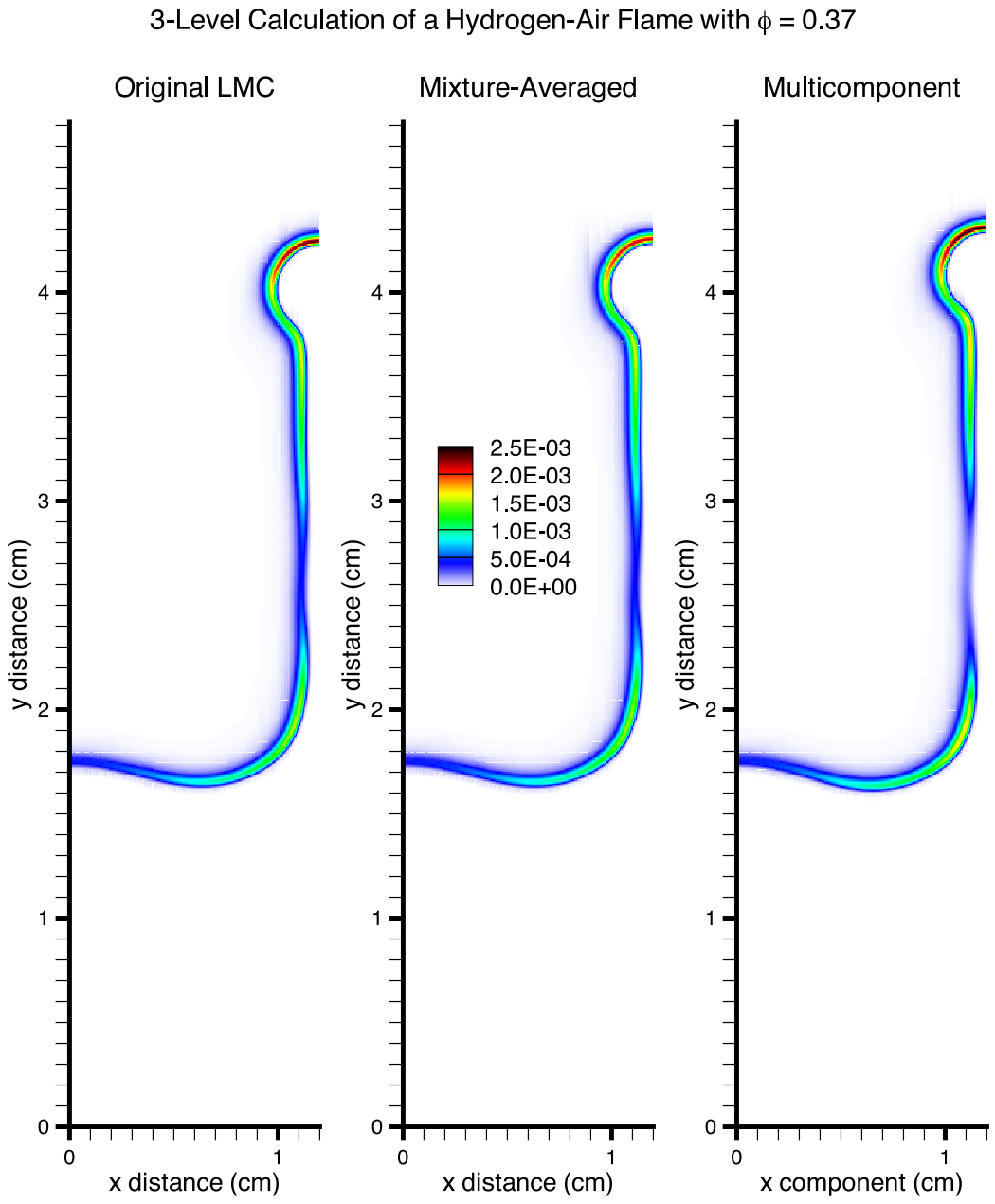

Figure 2: *Three calculations of the mole fraction of hydrogen, $X(\mathrm{H})$, in a freely-propagating, planar, hydrogen-air flame encountering a vortex.*

temporally resolved. Nevertheless, the shape of the flame is determined by convection from the vortical flow, so all three calculations show good agreement.

This problem shows some difference between the mixture-averaged and multicomponent models. In the multicomponent model $X(\mathrm{H})$ attains a maximum of 0.0025 whereas in the mixture-averaged model the value is no larger than 0.0022. See Table 3. The much larger time steps taken by the original LMC code accounts for the differences between the two mixture-averaged models. Decreasing the CFL limit from 0.80 to 0.14 decreases the time step of the original code from 22 to 4 microseconds and brings the peak $X(\mathrm{H})$ into agreement for both mixture-averaged models. Section 6.1 presents more information about time steps.

Table 3: *Centerline peak mole fraction of atomic hydrogen at* 0.01 *(sec) in the three calculations of a hydrogen-air flame encountering a vortex.*

diffusion model	solution algorithm	peak $X(\mathrm{H})$
mixture-averaged	original LMC	0.002424
mixture-averaged	Runge-Kutta	0.002198
multicomponent	Runge-Kutta	0.002484

6 Timing

6.1 Data

Table 4 shows information of the computer time required for (coarse-level) time steps in the two algorithms and two models, for both the 1D and 2D test cases.

Table 4: *(top) Time steps in the 1D calculations on the LBNL/HPCRD hive cluster. (bottom) Time steps in the 2D calculations on the NERSC bassi cluster. Values are determined by examining the final few time steps of the calculations.*

3-level, 2-processor calculations		one time step		
diffusion model	solution algorithm	size (micro s)	number	wall clock time (s)
mixture-averaged	original LMC	422	144	6.94
mixture-averaged	Runge-Kutta	1.76	28168	2.16
multicomponent	Runge-Kutta	1.76	28380	3.24

3-level, 8-processor calculations		one time step		
diffusion model	solution algorithm	size (micro s)	number	wall clock time (s)
mixture-averaged	original LMC	22.4	406	21.1 – 23.3
mixture-averaged	Runge-Kutta	0.96	9435	11.0 – 11.4
multicomponent	Runge-Kutta	0.94	9672	14.2 – 14.7

6.2 Performance Improvements

Two performance improvements are possible. First, the Runge-Kutta algorithm evaluates `rk_diffusion_operator` 5 times:

1,2 A peculiarity in the coding of the `getViscTerms` procedure results in it being called twice, once for flux divergences of the ρY_k and then again for flux divergence of ρh. Each of these calls to `getViscTerms` evaluates `rk_diffusion_operator`.

3,4 The `advance` procedure evaluates `rk_diffusion_operator` twice during the Runge-Kutta algorithm.

5 The `calc_divu` procedure calls `rk_diffusion_operator` once to obtain values for the evaluation of equation (9).

Evaluations 1, 2, and 3 are all applied to the same (old) state data, so by saving the output of `rk_diffusion_operator`, the number of calls could be reduced from 5 to 3 per time step.

Second, subroutines `FORT_RK_MIXTURE_AVERAGED` and `FORT_RK_MULTICOMPONENT` store state data in the natural `Fab` indexing order (i, j, n), where i and j specify the cell and n specifies the component. Consequently, the data for a given cell are not contiguous. Memory performance might be improved by transposing the data to an (n, i, j) ordering.

A Appendix. Notation

$c_{p,mix}$ Specific heat of the mixture, at constant pressure, per unit mass.

D/Dt Convective derivative, see Appendix B.

$\mathcal{D}_{k,\ell}$ Multicomponent diffusion coefficient for the k-th and ℓ-th species.

$\mathcal{D}_{k,mix}$ Mixture-averaged diffusion coefficient for the k-th species.

Δt Timestep.

Δx Mesh spacing.

h Mixture enthalpy per unit mass.

h_k Enthalpy per unit mass of the k-th species.

$\mathbf{j}_k$ Diffusive flux vector of the k-th species, either $\mathbf{j}_k^{(\mathrm{Mix})}$ or $\mathbf{j}_k^{(\mathrm{Multi})}$.

$\mathbf{j}_q$ Heat flux vector, either $\mathbf{j}_q^{(\mathrm{Mix})}$ or $\mathbf{j}_q^{(\mathrm{Multi})}$.

λ Thermal conductivity.

λ' Partial thermal conductivity.

$\bar{m}$ Mean molecular weight.

m_k Molecular weight of the k-th species.

N Spatial dimension of the problem, 2 or 3.

n Time step index.

ρ Mass density.

ρ_k Mass density of the k-th species, $\rho_k = \rho Y_k$.

$\dot{\rho}_k$ Rate of change in the mass density of the k-th species due to chemical reactions, $\dot{\rho}_k = m_k \dot{\omega}_k$

$\mathbf{S}$ Formula for $\nabla \cdot \mathbf{v}$, see section 3.4.

T Temperature.

t Time.

t_n Time at index n.

θ_k Thermal diffusion coefficient for the k-th species.

$\mathbf{v}$ Velocity.

X_k Mole fraction of the ℓ-th species.

Y_k Mass fraction of the k-th species.

$\dot{\omega}_k$ In the notation of [6], molar production rate of the k-th species (mol / vol sec).

B Appendix. Convective Derivative

The convective derivative (also known as the substantive derivative, the material derivative, the Lagrangian derivative, and the derivative along particle paths) is

$$\frac{D\phi}{Dt} = \frac{\partial \phi}{\partial t} + \mathbf{v} \cdot \nabla \phi$$

The continuity equation

$$\frac{\partial \rho}{\partial t} + \nabla \cdot \rho \mathbf{v} = 0$$

implies two identities for the convective derivative. First, it removes the time derivative from the convective derivative of density,

$$\begin{aligned} \frac{D\rho}{Dt} &= \frac{\partial \rho}{\partial t} + \mathbf{v} \cdot \nabla \rho \\ &= \frac{\partial \rho}{\partial t} + (\nabla \cdot \rho \mathbf{v} - \nabla \rho \cdot \mathbf{v} - \rho \nabla \cdot \mathbf{v}) + \mathbf{v} \cdot \nabla \rho \\ &= -\rho \nabla \cdot \mathbf{v} \,. \end{aligned} \tag{10}$$

Second, it places the convective derivative into equations (1) and (2),

$$\begin{aligned}\frac{\partial \rho\phi}{\partial t} + \nabla \cdot \rho\phi\mathbf{v} &= \phi\,\frac{\partial \rho}{\partial t} + \rho\,\frac{\partial \phi}{\partial t} + \rho\mathbf{v}\cdot\nabla\phi + \phi\nabla\cdot\rho\mathbf{v} \\ &= \rho\,\frac{\partial \phi}{\partial t} + \rho\mathbf{v}\cdot\nabla\phi \\ &= \rho\,\frac{D\phi}{Dt}\ .\end{aligned} \tag{11}$$

C Appendix. Temperature Equation

The mixture and species enthalpies are related by

$$h = \sum_k h_k Y_k\ .$$

In a Newtonian fluid the enthalpies are functions only of pressure and temperature, and in our case pressure is constant. Differentiating this equation and using the chain rule gives

$$\begin{aligned}\frac{Dh}{Dt} &= \sum_k \frac{Dh_k}{Dt}\,Y_k + \sum_k h_k\,\frac{DY_k}{Dt} \\ &= \sum_k \frac{\partial h_k}{\partial T}\,\frac{DT}{Dt}\,Y_k + \sum_k h_k\,\frac{DY_k}{Dt}\end{aligned}$$

Multiplying by ρ, and since

$$\sum_k \frac{\partial h_k}{\partial T}\,Y_k = c_{p,mix}\ ,$$

and by equations (1) and (2), therefore

$$-\nabla\cdot\mathbf{j}_q = \rho c_{p,mix}\frac{DT}{Dt} + \sum_k h_k\left(\dot\rho_k - \nabla\cdot\mathbf{j}_k\right)$$

which rearranges to

$$\rho c_{p,mix}\frac{DT}{Dt} = -\nabla\cdot\mathbf{j}_q - \sum_k h_k\left(\dot\rho_k - \nabla\cdot\mathbf{j}_k\right)\ . \tag{12}$$

For the mixture-averaged model this equation simplifies to

$$\rho c_{p,mix}\frac{DT}{Dt} = \nabla\cdot\lambda\nabla T + \sum_k \rho\mathcal{D}_{k,mix}\nabla h_k\cdot\nabla Y_k - \sum_k h_k\dot\rho_k\ ,$$

which is the formula given by Day and Bell [3, p. 540, eqn. 12].

D Appendix. Alternate Enthalpy Equation

The LMC code currently uses a Crank-Nicolson method to update h from diffusion, so the ∇T term in equation (5) is replaced by another term involving ∇h. This equation is not needed for the implementation of the Runge-Kutta method. Nevertheless, the following derivation is given for completeness. Since enthalpies are functions only of T for an ideal gas, and from

$$h = \sum_k h_k Y_k \qquad \frac{\partial h_k}{\partial T} = c_{p,k} \qquad c_{p,mix} = \sum_k c_{p,k} Y_k$$

follow

$$\begin{aligned} \nabla h &= \sum_k h_k \nabla Y_k + \sum_k (\nabla h_k) Y_k \\ &= \sum_k h_k \nabla Y_k + \sum_k c_{p,k} Y_k \nabla T \\ &= \sum_k h_k \nabla Y_k + c_{p,mix} \nabla T \end{aligned}$$

therefore

$$\nabla T = \frac{1}{c_{p,mix}} \nabla h - \sum_k \frac{h_k}{c_{p,mix}} \nabla Y_k \,. \tag{13}$$

Equation (13) may be substituted into either of the models for heat flux, equations (5) or (6), for use in the enthalpy equation (2). For the mixture-averaged model, this gives equation (11) in Day and Bell [3],

$$\frac{\partial \rho h}{\partial t} + \nabla \cdot \rho h \mathbf{v} = \nabla \cdot \frac{\lambda}{c_{p.mix}} \nabla h + \sum_k \nabla \cdot h_k \left(\rho \mathcal{D}_{k,mix} - \frac{\lambda}{c_{p,mix}} \right) \nabla Y_k \,.$$

For the multicomponent model, the formula is more complicated because there are ∇T terms in the diffusive flux vectors.

References

[1] J. B. Bell, M. S. Day, J. F. Grcar, and M. J. Lijewski. Active control for statistically stationary turbulent premixed flame simulations. *Comm. App. Math. Comput. Sci.*, 1(1):29–52, November 2005.

[2] H. Bongers and L. P. H. de Goey. The effect of simplified transport modeling on the burning velocity of laminar premixed flames. *Combust. Sci. Technol.*, 175:1915–1928, 2003.

[3] M. S. Day and J. B. Bell. Numerical simulation of laminar reacting flows with complex chemistry. *Combust. Theory Modelling*, 4:535–556, 2000.

[4] A. Ern and V. Giovangigli. EGLIB: A General-Purpose Fortran Library for Multicomponent Transport Property Evaluations. Technical report, 2004. Version 3.4.

[5] V. Giovangigli. *Multicomponent Flow Modeling*. Birkhäuser, Boston, 1999.

[6] R. J. Kee, R. M. Ruply, E. Meeks, and J. A. Miller. Chemkin-III: A FORTRAN chemical kinetics package for the analysis of gas-phase chemical and plasma kinetics. Technical Report SAND96-8216, Sandia National Laboratories, Livermore, 1996.